Chahan MOVSESSIAN
Nicolas GUIBLIN
Paul ZANAGLIA
Quentin SCORDO

Le 15/06/2023
Université Côte d'Azur
Master 1 Informatique

Rapport du projet de Big_Data

Utilisation du moteur NoSQL : Voldemort

Professeur:
Gabriel MOPOLO

Chapitre 1 : Choix du sujet

Nous avons décidé de reprendre notre sujet du semestre précédent, qui est : **la gestion d'un Zoo**. Ce choix a été particulièrement enrichissant du fait de la multitude d'informations et de données que nous pouvions manipuler et créer. Les différentes facettes de la gestion d'un zoo, de la gestion quotidienne des animaux à la coordination des ressources humaines en tant que clients et employés.

Notre base de données comprendra les tables suivantes : "animal", "employé", "département", "enclos", "soin", "visiteur" et "ticket". Chacune de ces tables aura des informations spécifiques, telles que l’identifiant de l'animal, les informations sur les employés, les secteurs de travail, les informations sur les enclos, les soins apportés aux animaux, les informations sur les visiteurs et les tickets achetés. L'application devra aussi permettre la gestion et la mise à jour de ces informations.

Chapitre 2 : MCD MERISE

· Le dictionnaire de données MERISE

La première est la table Animal elle possède 9 propriétés :

• ANIMNO : c'est l'identifiant unique de l'animal permettant de différencier chaque animal, il est sous format number(8) et ne peut pas être NULL

• ANAME : c'est le nom sous format varchar2. Il ne peut pas être NULL

• ESPECE : c'est l'espèce de l'animal sous format varchar2 également. Il ne peut pas être NULL

• SEXE : C'est un Varchar2 limité à un seul caractère entre « F » et « M », cela permet de savoir si notre animal est une femelle ou un mâle. Il ne peut pas être NULL

• DIET : il permet de savoir quel régime alimentaire l'animal suit, c'est un Varchar2 limité à trois propositions :carnivore, herbivore ou omnivore. Il ne peut pas être NULL

• DATE_NAISS : c'est la date de naissance permettant de savoir l'âge de l'animal, elle est donc sous format date, elle ne pas être strictement supérieur à la date d'arrivé et ne peut pas être NULL.

• DATE_ARRIVE, qui sont et la date d'arrivé de l'animal en question dans le zoo car il y a souvent des transferts d'animaux, elle est donc sous format date également. Elle ne peut pas être inférieur à la date de création du zoo c'est-à-dire le 01/01/2000 et ne peut pas être NULL.

• POIDS : c'est le poids de l'animal en kg, il est sous format number. Il doit être supérieur à 0 et ne peut pas être NULL

• TAILLE : c'est la taille de l'animal en cm, elle est sous format number. Il doit être supérieur à 0 et ne peut pas être NULL

La seconde table est la table Enclos, elle possède 4 propriétés :

• ENCNO : c'est l'identifiant unique de l'enclos, il est sous format number. Il ne peut pas être NULL

• ENCNAME : c'est le nom de l'enclos il est sous format varchar2 et pour le choisir on met simplement le nom des animaux qui sont dedans exemple : si on a un enclos à loup et renard son nom sera « Loup et Renard ». Il ne peut pas être NULL

• CAPACITE : c'est la capacité d'animaux que l'enclos peut avoir, c'est donc un format number. Il ne peut pas être NULL

• TAILLE : c'est le nombre de m2 que fait l'enclos en format number. Il ne peut pas être NULL

La troisième table est la table SOIN, elle possède 3 propriétés :

• SOINNO : c'est l'identifiant unique du soin permettant de différencier chaque soins, il est sous format number(8) et ne peut pas être NULL

• SNAME : le nom du soin, si il s'agit d'une vaccination, d'une consultation etc. Elle est sous format Varchar2 et ne peut pas être NULL

• DATE_SOIN : c'est la date où aura lieu le soin en question sous format date. Elle ne peut pas être NULL et doit être supérieur au 01/01/2000.

La quatrième table est la table EMPLOYE, elle possède 8 propriétès :

• EMPNO : c'est l'identifiant unique de l'employé, il est sous format number(8) et ne peut pas être NULL

• ENAME : Il s'agit du nom de famille de l'employé en lettre capitales et non null. C'est un varchar2 et ne peut pas être NULL

• PRENOMS : C'est la liste des prénoms de l'employé donc elle est sous format varray. L'employé ne peut pas avoir zéro prénom et ne peut en avoir plus de 4.

• CV : C'est le CV d'embauche de l'employé sous format CLOB

• JOB : C'est la fonction de l'employé sous format Varchar2. Il doit forcément appartenir à ces fonctions : 'Directeur', 'Secretaire', 'Community Manager', 'Comptable', 'Vendeur Billetterie', 'Chef equipe surface', 'Technicien de surface', Guide', 'Guide scolaire','Veterinaire', 'Soignant', 'Logisticien', 'Technicien' et ne peut donc pas être NULL

• SAL : C'est le salaire par mois de l'employé, il s'agit d'un Number compris entre 1500 et 15000 donc non NULL

• DATE_NAISS : date de naissance de l'employé sous format date et non NULL

• DATE_EMB : date d'embauche de l'employé ne peut pas être inférieur à la date de naissance et ne peut pas être NULL, il s'agit donc d'un format date.

La cinquième table est celle des DEPARTEMENT, elle possède 2 propriétés :

• DPTNO : l'identifiant unique du département, sous format Number compris entre 1000 et 9999, et non NULL

• DNAME : Le nom du département en Varchar2 non NULL et qui doit forcément être unique et parmi : Administratif, Billetterie, Entretien, Guide et Service des soins

La sixième table est celle des VISITEURS qui possède 6 propriétés :

• VISNO : c’est l’identifiant unique du visiteur, il est sous format number(8) et ne peut pas être NULL

• VISNAME : Il s’agit du nom de famille du visiteur en lettre capitales et non null. C’est un varchar2

• PRENOMS : C’est la liste des prénoms du visiteurs donc elle est sous format varray. L’employé ne peut pas avoir zéro prénom et ne peut en avoir plus de 4.

• VILLE : Il s’agit du nom de la ville où vit le visiteur, elle est sous format Varchar2 et ne peut pas être NULL

• AGE : Il s’agit de l’âge du visiteur non NULL compris entre 0 et 120 et sous format Number

• DATE_VISITE : c’est la date de visite sous format date. Elle ne peut pas être NULL et doit forcément être supérieur à la date de création du zoo c’est-à-dire le 01/01/2000

Enfin la septième table est la table TICKET et elle possède 4 propriétés :

• TICKETNO : est l’identifiant unique du ticket en format Number et non NULL • DATE_ACHAT : date à laquelle le visiteur à acheter le ticket en quesion. Sous format date supérieur au 01/01/2000 et non NULL

• CATEGORIE : est un varchar2 inclus dans cette liste : 'enfant', 'etudiant', 'adulte', 'retraité', 'groupe'

• TARIF : il s’agit du tarif du ticket, c’est un Number compris entre 9.99 et 19.99, il est non NULL

· La description textuelle des associations

Les associations entre les entités dans notre base de données sont :

• La table "animal" est associée à la table "enclos" via une relation un-à-plusieurs, car un enclos peut contenir plusieurs animaux, mais un animal ne peut être contenu dans qu'un seul enclos.

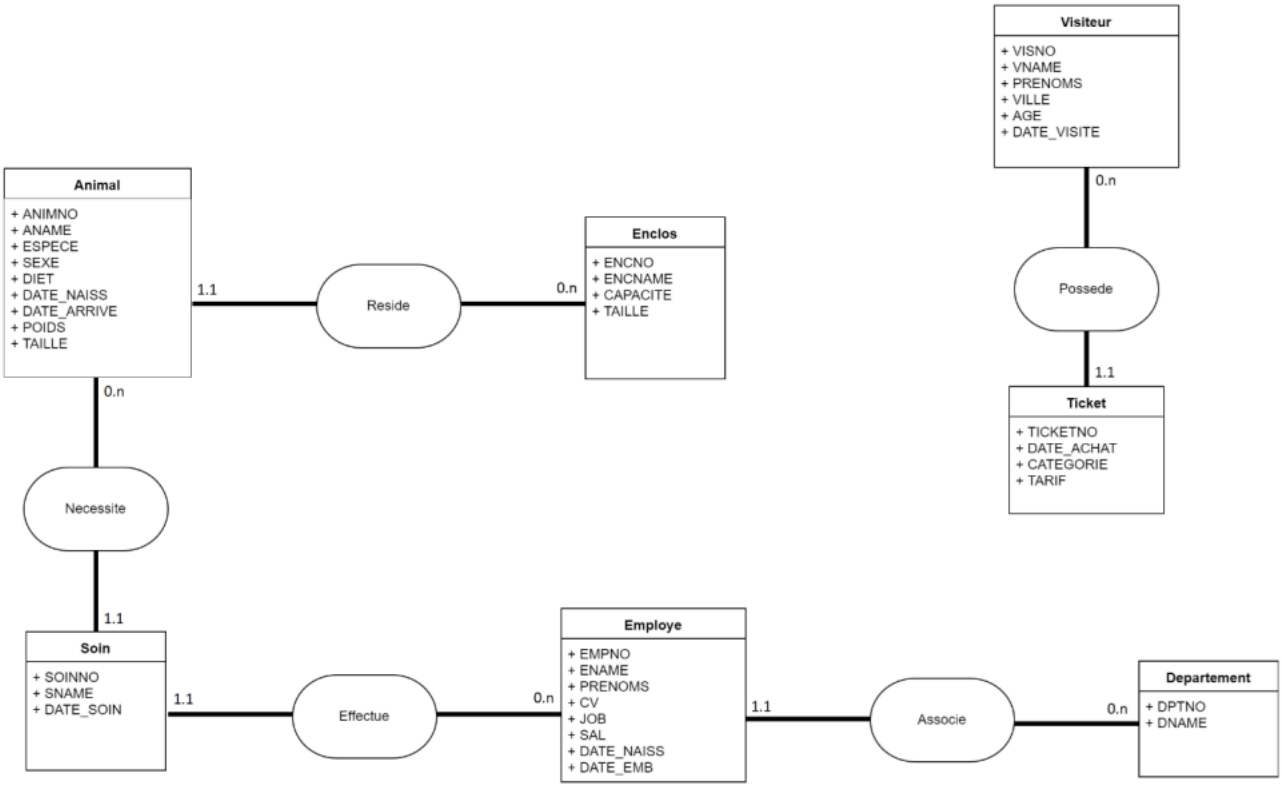
• La table "animal" est associée à la table "soin" via une relation plusieurs-à-plusieurs, car un animal peut recevoir plusieurs soins et un soin peut être apporter différents animaux.

• La table "employé" est associée à la table "département" via une relation un-àplusieurs, car un employé appartient à un seul département et un département peut être associé à plusieurs employés.

• La table "Employé" est associée à la table "soin" via une relation un-à-plusieurs, car un employé peut apporter plusieurs soins et un soin ne peut être apporter que par un seul employé.

• La table "visiteur" est associée à la table "ticket" via une relation un-à-plusieurs, car un visiteur peut acheter plusieurs tickets et un ticket peut être associé à un seul visiteur. Ces associations permettent de gérer les informations du zoo de manière organisée et efficace, en créant des liens logiques entre les différentes parties de la base de données.

· La définition du Modèle Entité-Association MERISE



Chapitre 3 :

· Spécification des modèles de documents à mettre dans chaque collection

Chaque entité de notre modèle de données - Visitor, Ticket, Employee, Department, Treatment, Animal, Enclosure - est stockée dans son propre "store" dans Voldemort. Cela est similaire à la façon dont les collections sont utilisées dans MongoDB ou les tables dans une base de données relationnelle.

Chaque instance d'une entité est stockée comme une paire clé-valeur. La clé est l'ID de l'entité et la valeur est une représentation JSON de l'entité.

Un visiteur est stocké comme suit :

Clé : "1"
Valeur : "{"VNom":"Reely","VPrenom":"Silva","VVille":"Las Toscas","VAge":86,"VDateVisite":"2022-12-26","_id":"1"}"

Un employé est stocké comme suit :

Clé : "1"
Valeur : "{"empPrenom":"Brel","empNom":"Jacques","empCV":"https://about.com/velit/nec/nisi/vulputate.jpg?tortor\u003dnunc\u0026risus\u003drhoncus\u0026dapibus\u003ddui\u0026augue\u003dvel\u0026vel\u003dsem\u0026accumsan\u003dsed\u0026tellus\u003dsagittis","empJob":"Civil Engineer","empSal":2300.0,"empDateNaiss":"1981-06-23","empDateEmb":"2011-08-20","dept":213,"_id":"1"}"

Un animal est stocké comme suit :

Clé : "1"
Valeur : "{"AnimName":"Ambre","Espece":"Elephantidae","Sexe":"F","Diet":"Carnivore","DateNaiss":"2019-11-18","DateArriv":"2011-01-18","Poids":346,"Taille":131,"Enclos":567,"_id":"1"}"

Un ticket est stocké comme suit :

Clé : "1"
Valeur :"{"dateAchat":"2020-12-12","categorie":"Enfant","tarif":50,"appartenance":330,"_id":"1"}"

Un département est stocké comme suit :

Clé : "1" Administratif
Valeur : "{"Dname":"Administratif","_id":"1"}"

Un soin est stocké comme suit :

Clé : "1"
Valeur : "{"SoinName":"Toilettage","DateSoin":"2020-12-12","AnimId":619,"EmpId":757,"_id":"1"}"

Un enclos est stocké comme suit :

Clé : "1"

Valeur : "{"EncName":"Enclos des lions","Capacite":96,"Taille":130,"_id":"1"}"

Cela signifie que chaque entité est stockée de manière indépendante. Les relations entre les entités sont gérées au niveau de l'application. Par exemple, si un ticket appartient à un visiteur, l'ID du visiteur est stocké dans le ticket. Pour obtenir les informations sur le visiteur à partir du ticket, nous récupérons d'abord l'ID du visiteur à partir du ticket, puis nous utilisons cet ID pour récupérer les informations du visiteur à partir du store des visiteurs.

Sur le plan du code Java, chaque entité est représentée par une classe Java distincte. Ces classes contiennent des champs pour chaque attribut de l'entité, ainsi que des méthodes pour obtenir et définir ces attributs (getters et setters). Par exemple, la classe Visitor a des champs pour l'ID, le prénom, le nom, l'âge, la date de naissance et l'e-mail du visiteur, ainsi que des méthodes pour obtenir et définir ces attributs.

▪ Spécification des classes et des méthodes JAVA

Dans ce projet, nous avons défini plusieurs classes en Java pour représenter les différentes entités de notre application. Chaque classe a des attributs correspondant aux champs de l'entité et des méthodes pour manipuler ces attributs.

Chaque classe a des méthodes pour effectuer les opérations CRUD sur les instances de l'entité :

- **create(String json)**: Cette méthode prend une chaîne JSON en entrée, la convertit en une instance de l'entité et l'insère dans la base de données. Si un document avec le même ID existe déjà, il est remplacé par le nouveau document.
- **read()**: Cette méthode récupère le document avec l'ID de l'instance de l'entité de la base de données et met à jour les attributs de l'instance avec les valeurs du document.
- **update(String json)**: Cette méthode prend une chaîne JSON en entrée, la convertit en une instance de l'entité et met à jour le document correspondant dans la base de données avec les nouvelles valeurs.
- **delete()**: Cette méthode supprime le document avec l'ID de l'instance de l'entité de la base de données.

Chapitre 4 : Complément sur le moteur NoSql Voldemort

- Modèles de données supportés :

Voldemort est une base de données clé-valeur distribuée, ce qui signifie qu'elle stocke des données sous la forme de paires clé-valeur. Ces valeurs peuvent être des valeurs binaires (c'est-à-dire des données brutes), du texte ou des objets sérialisés.

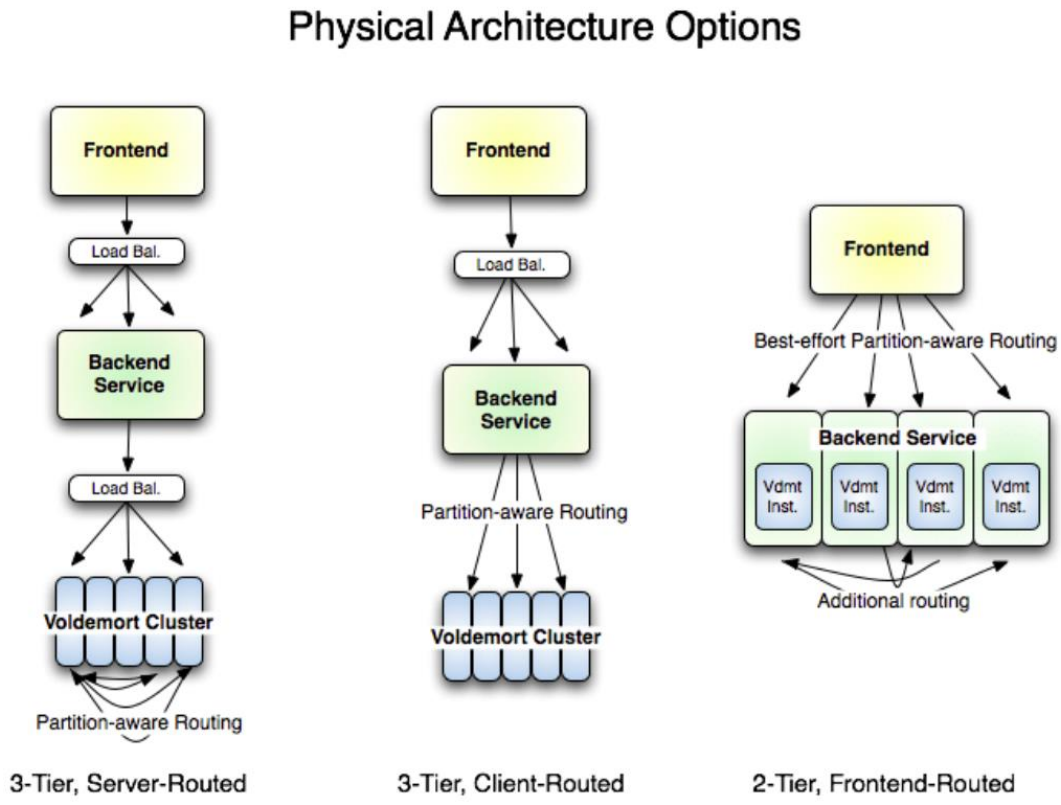
- Procédure d'installation du moteur et des utilitaires :

En ce qui concerne l'installation de Voldemort, nous l'avons effectuée en clonant le répertoire git, qui contient le code source du programme et donc l'accessibilité au serveur, nous l'avons compilé à l'aide de Gradle, un outil d'automatisation de build. Nous avons pris connaissances de la documentation détaillée sur l'installation qui est disponible sur le dépôt GitHub officiel de Voldemort.

- Architecture du moteur NoSql :

L'architecture de Voldemort est décentralisée, ce qui signifie qu'il n'y a pas de point central de défaillance. Chaque nœud dans la base de données fonctionne indépendamment des autres, ce qui permet une réplication et un partitionnement automatique des données. Cela contribue à la haute disponibilité et à la tolérance aux pannes de la base de données.

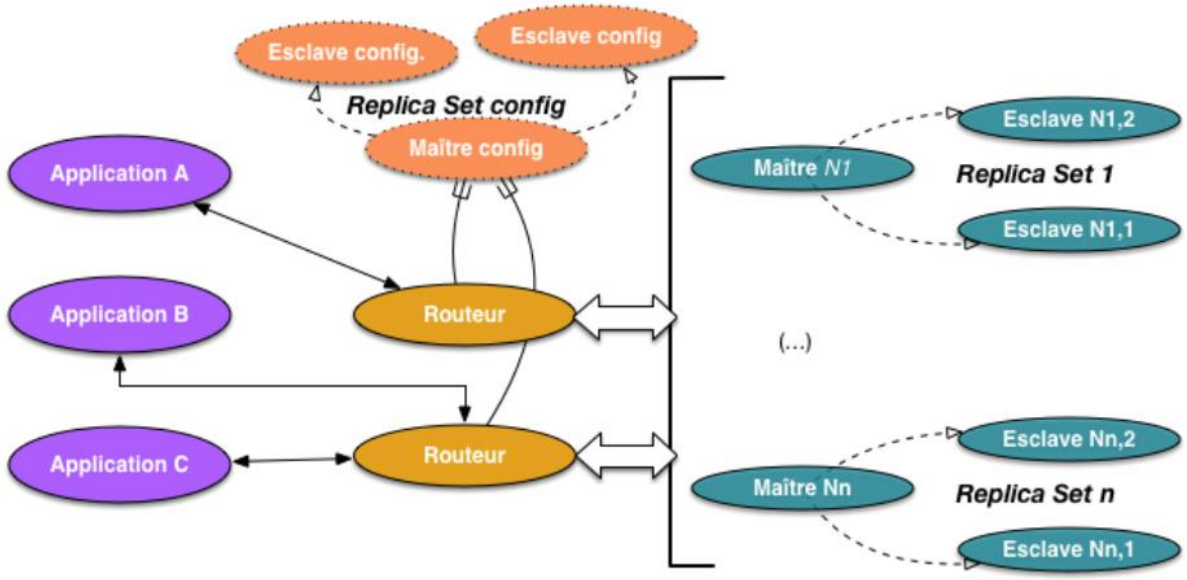
Scéma d'un projet



physique sur l'architecture Voldemort par Jesús Sánchez Cuadrado

- Méthode de partitionnement :

Voldemort utilise le partitionnement basé sur la clé de hachage consistent. Cela signifie que les données sont réparties entre plusieurs nœuds en fonction de la valeur de hachage de la clé. Cela permet une distribution équilibrée des données et une recherche efficace.
Cela nous à fortement contribué à choisir ce moteur de données NoSQL car nous avions vraiment envie de travailler avec une méthode différente que celle utilisé par MongoDB qui est le clé/document.



Scéma de partitonnement d'un moteur NoSQL Voldemort par bdpedia

- Méthode de réplication :

Voldemort réplique les données sur plusieurs serveurs pour assurer la disponibilité des données en cas de défaillance d'un ou plusieurs nœuds. Cette réplication est automatique, ce qui signifie que le système gère la réplication sans intervention humaine.

- Montée en charge :

Voldemort peut gérer une montée en charge horizontale, ce qui signifie qu'il peut ajouter plus de nœuds pour gérer une augmentation de la charge de travail. Cette capacité à augmenter la capacité de traitement par l'ajout de nœuds rend Voldemort hautement évolutif.
C'est une des raisons pour laquelle Voldemort est encore utilisé aujourd'hui.

- Gestion du ou des caches mémoire :

Voldemort utilise un cache en mémoire pour améliorer les performances de lecture des données. Les éléments les plus fréquemment utilisés sont conservés en mémoire pour un accès plus rapide, réduisant ainsi le temps nécessaire pour retrouver des données fréquemment utilisées.

Chapitre 5 : Génération automatique des objets

Pour la génération automatique des données, nous avons utilisé le site Mockaroo. Ce site offre la possibilité de générer des données dans divers formats, dont JSON et CSV, et permet de personnaliser les données générées en fonction de nos besoins. Nous avons généré plusieurs milliers d'objets .

Une fois les données générées, nous avons créé une classe Java DatabaseBuilder pour insérer ces données dans nos bases de données. Cette classe utilise les opérations CRUD que nous avons définies dans nos classes d'entités pour ajouter les données générées à la base de données. De plus, nous avons également implémenté une méthode pour supprimer toutes les données de la base de données, ce qui peut être utile pour réinitialiser la base de données à un état vide.

Exemple génération Employés :